

# TEMA: POO UTILIZANDO JAVA. PARTE II

---

Taller de Programación.

Módulo: Programación Orientada a Objetos

# Instanciar e iniciar objeto

- Hasta ahora, nuestro main ...

```
public class Demo01Libro {  
  
    public static void main(String[] args) {  
        Libro libro = new Libro();  
        libro.setTitulo("Java: A Beginner's Guide");  
        libro.setEditorial("Mcgraw-Hill");  
        libro.setAñoEdicion(2014);  
        libro.setPrimerAutor("Herbert Schildt");  
        libro.setISBN("978-0071809252");  
        libro.setPrecio(21.72);  
        ...  
    }  
}
```

Generar una clase para representar libros. Un Libro se caracteriza por: título, nombre del primer autor, editorial, año de edición, ISBN, precio

El libro debe saber:

- Devolver el valor de cada atributo.
- Modificar el valor de cada atributo.
- Devolver su representación en formato String.  
Repr. *"Java: A Beginner's Guide por Herbert Schildt - 2014 - ISBN: 978-0071809252"*

Libro
titulo, primerAutor, editorial, añoEdicion, ISBN, precio
<code>String getTitulo()</code> ... <code>double getPrecio()</code> <code>void setTitulo(String unTitulo)</code> ... <code>void setPrecio(double unPrecio)</code> <code>String toString()</code>

# Declaración de constructores.

- Se ejecuta tras alocar el objeto e inicializar las v.i. (por defecto o explícitamente).
- Objetivo: inicialización de v.i.
- Sintaxis

```
public NombreClase( lista de parámetros formales ) {  
    /* Código */  
}
```

- Si la clase no declara ningún constructor, Java incluye uno sin parámetros y sin código (*constructor nulo*).
- Instanciación de objeto:

```
NombreClase objeto= new NombreClase(lista de parámetros actuales);
```

Ejemplo (Hasta ahora) Libro miLibro = new Libro(); *//Invoca al constructor nulo.*

## Declaración de constructores. Ejemplo.

```
public class Libro {  
    private String titulo;  
    private String primerAutor;  
    private String editorial;  
    private int añoEdicion;  
    private String ISBN;  
    private double precio;
```

```
    public Libro( String unTitulo, String unaEditorial,  
                 int unAñoEdicion, String unPrimerAutor,  
                 String unISBN, double unPrecio){  
        titulo = unTitulo;  
        editorial = unaEditorial;  
        añoEdicion= unAñoEdicion;  
        primerAutor = unPrimerAutor;  
        ISBN = unISBN;  
        precio = unPrecio;  
    }  
  
    ....  
  
}
```

## Declaración de constructores. Ejemplo.

- Ejemplo instanciación (en main)

```
Libro libro1= new Libro( "Java: A Beginner's Guide", "Mcgraw-Hill",  
                        2014, "Herbert Schildt",  
                        "978-0071809252", 21.72);
```

- ¿Funciona ahora? Libro libro = new Libro();

Si el programador generó un constructor,  
Java no incluye el constructor nulo.

# Declaración de constructores. Sobrecarga. Ejemplo.

- Puede haber varios constructores para la clase (sobrecarga).
- Java identifica cuál está siendo invocado por el número y tipo de sus parámetros.
- *Por defecto quiero que el libro tenga año de edición 2015 y precio 100 => Otro constructor*

```
public class Libro {  
    private String titulo;  
    private String primerAutor;  
    private String editorial;  
    private int añoEdicion;  
    private String ISBN;  
    private double precio;  
  
    public Libro( String unTitulo, String unaEditorial,  
int unAñoEdicion, String unPrimerAutor, String unISBN, double unPrecio){  
        titulo = unTitulo;  
        editorial = unaEditorial;  
        añoEdicion= unAñoEdicion;  
        primerAutor = unPrimerAutor;  
        ISBN = unISBN;  
        precio = unPrecio;  
    }  
}
```


```
    public Libro( String unTitulo, String unaEditorial, String  
unPrimerAutor, String unISBN){  
        titulo = unTitulo;  
        editorial = unaEditorial;  
        añoEdicion= 2015;  
        primerAutor = unPrimerAutor;  
        ISBN = unISBN;  
        precio = 100;  
    }  
  
    public Libro(){  
  
    }  
    ...  
}
```

3 constructores distintos

Libro.java

# Declaración de constructores. Sobrecarga. Ejemplo.

```
public class Demo01ConstructoresLibro {  
    public static void main(String[] args) {  
        Libro libro1= new Libro( "Java: A Beginner's Guide", "Mcgraw-Hill", 2014,  
                                "Herbert Schildt", "978-0071809252", 21.72);  
        Libro libro2= new Libro("Learning Java by Building Android Games",  
                                "CreateSpace Independent Publishing",  
                                "John Horton", "978-1512108347");  
        System.out.println(libro1.toString());  
        System.out.println(libro2.toString());  
        System.out.println("Precio del libro2: " +libro2.getPrecio());  
        System.out.println("Año edición del libro2: " +libro2.getAñoEdicion());  
        Libro libro3= new Libro();  
    }  
}
```



¿Funciona?

## Interacción entre objetos (Ejercicio 3)

- Normalmente un Prog. OO tiene objetos de distintas clases.
- Los objetos cooperan (enviándose mensajes) para llevar a cabo una tarea común ...
- **Antes:** nuestros libros consideraban el nombre del primer autor (String).
- **Ahora:** quiero que el libro conozca del primer autor nombre, biografía, etc
- ¿Qué estrategia seguir?

~~¿Incluir al libro todos los datos del primer autor  
... y comportamiento?~~

¿Hacer que el libro conozca a un obj. autor?





# Interacción entre objetos (Ejercicio 3)

- Un libro conoce a su autor (obj).

Diagrama de clases

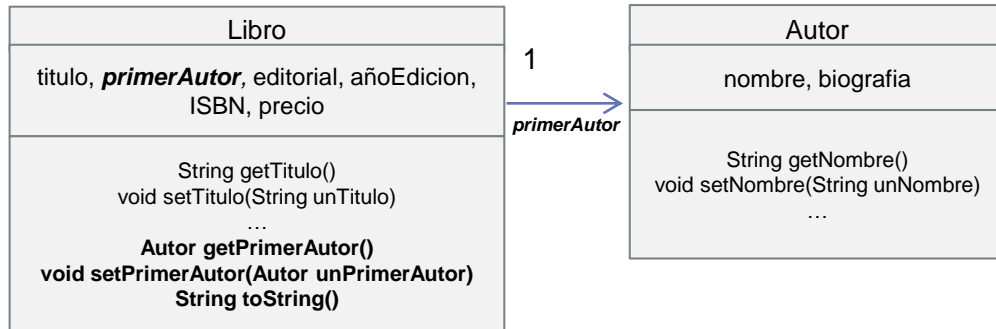
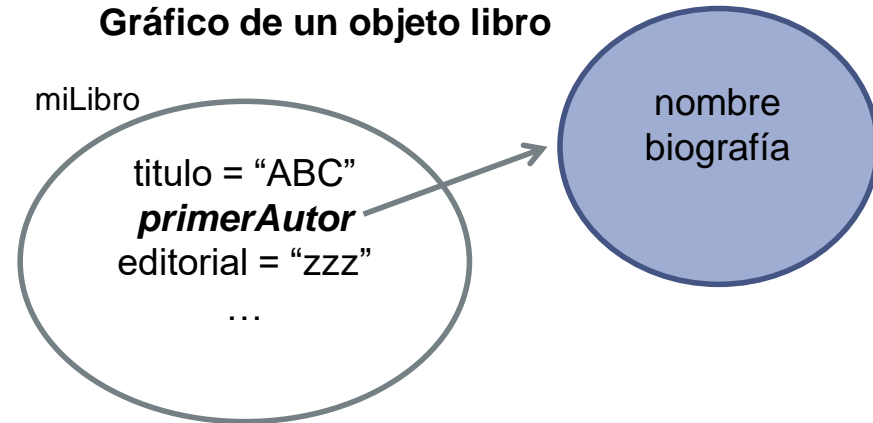


Gráfico de un objeto libro



- Modificaciones en el código (carpeta tema4)
  - Generar la clase Autor
  - Modificar la clase Libro
  - Modificar el Programa Principal

# Interacción entre objetos (Ejercicio 3)

```
public class Libro {
    private String titulo;
    private String primerAutor;
    private String editorial;
    private int añoEdicion;
    private String ISBN;
    private double precio;

    public Libro( String unTitulo, String unaEditorial,
        int unAñoEdicion, String unPrimerAutor, String unISBN, double unPrecio){
        titulo = unTitulo;
        editorial = unaEditorial;
        añoEdicion= unAñoEdicion;
        primerAutor = unPrimerAutor;
        ISBN = unISBN;
        precio = unPrecio;
    }

    public String getTitulo(){
        return titulo;
    }

    public void setTitulo(String unTitulo){
        titulo = unTitulo;
    }
}
```

primerAutor ahora será instancia de clase Autor

constructor ¿qué debe recibir?

getPrimerAutor  
¿qué debe devolver?

setPrimerAutor  
¿qué debe recibir?

¿cómo obtengo el nombre del primerAutor?

```
...

public String getPrimerAutor() {
    return primerAutor;
}

public void setPrimerAutor(String unPrimerAutor){
    primerAutor=unPrimerAutor;
}

@Override
public String toString(){
    String aux;
    aux= titulo + " por " + primerAutor + " - " +
        añoEdicion + " - " + " ISBN: " + ISBN;
    return ( aux);
}
}
```

¿Cómo instancio un libro en el Prog. Ppal?

# Interacción entre objetos (Ejercicio 3)

Diagrama de clases

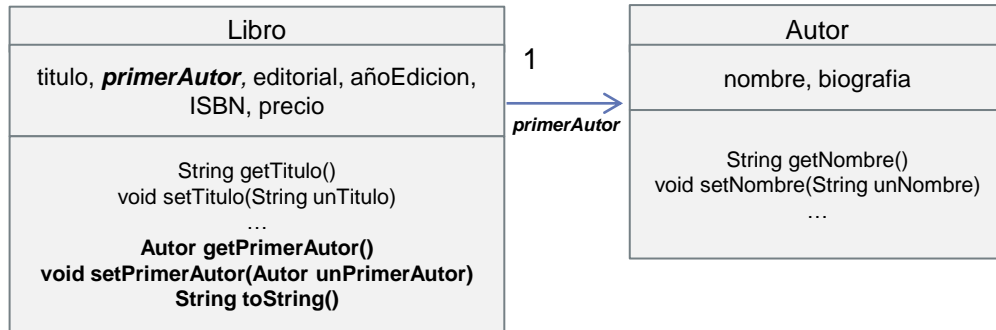
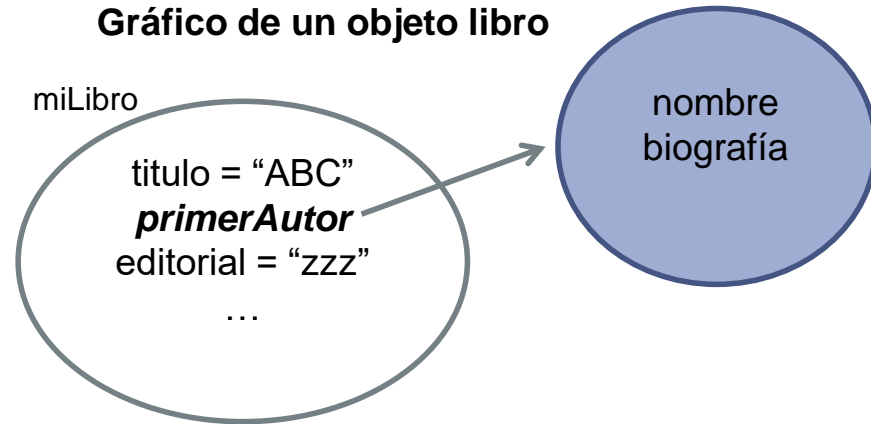


Gráfico de un objeto libro



- En prog. ppal...dado el objeto **miLibro** ... ¿qué pasos sigo para imprimir el nombre de su autor?
- Pido al objeto **miLibro** que me devuelva su autor ... ¿cómo?
- Pido al autor que me devuelva su nombre y lo imprimo ... ¿cómo?

## La referencia this. Uso.

- Dentro de una clase ... ¿Puedo disparar la ejecución de un método **X** desde otro método **Y**?
- ¿Utilidad? Ejemplo: añadir métodos al libro para obtener su IVA y su precio final con IVA.

```
public class Libro {  
    private String titulo;  
    private String primerAutor;  
    private String editorial;  
    private int añoEdicion;  
    private String ISBN;  
    private double precio;  
  
    public Libro( String unTitulo, String unaEditorial,  
        int unAñoEdicion, String unPrimerAutor,  
        String unISBN, double unPrecio){  
        titulo = unTitulo;  
        editorial = unaEditorial;  
        añoEdicion= unAñoEdicion;  
        primerAutor = unPrimerAutor;  
        ISBN = unISBN;  
        precio = unPrecio;  
    }  
    ...  
}
```

```
...  
  
    public double getMontoIva(){  
        return precio*0.21;  
    }  
  
    public double getPrecioFinalConIva(){  
        return precio +      ???  
    }  
}
```

Tengo un método que calcula el IVA...  
¿Cómo disparo su ejecución?

# La referencia this. Uso.

- Dentro de una clase ... ¿Puedo disparar la ejecución de un método **X** desde otro método **Y**?
- ¿Utilidad? Ejemplo: añadir métodos al libro para obtener su IVA y su precio final con IVA.

```
public class Libro {  
    private String titulo;  
    private String primerAutor;  
    private String editorial;  
    private int añoEdicion;  
    private String ISBN;  
    private double precio;  
  
    public Libro( String unTitulo, String unaEditorial,  
        int unAñoEdicion, String unPrimerAutor,  
        String unISBN, double unPrecio){  
        titulo = unTitulo;  
        editorial = unaEditorial;  
        añoEdicion= unAñoEdicion;  
        primerAutor = unPrimerAutor;  
        ISBN = unISBN;  
        precio = unPrecio;  
    }  
    ...  
}
```

...

```
public double getMontoIva(){  
    return precio*0.21;  
}
```

```
public double getPrecioFinalConIva(){  
    return precio + this.getMontoIva();  
}
```

}

Poniendo **this.nombreMétodo (parámetros)**

El objeto que está ejecutando (**this**) se enviará un mensaje a sí mismo.

El método a ejecutar se busca a partir de la clase de la cual es instancia el objeto.

## La referencia this. Uso.

- Dentro de una clase ... ¿Puedo disparar la ejecución de un método **X** desde otro método **Y**?
- ¿Utilidad? Ejemplo: añadir métodos al libro para obtener su IVA y su precio final con IVA.

```
public class Libro {  
    private String titulo;  
    private String primerAutor;  
    private String editorial;  
    private int añoEdicion;  
    private String ISBN;  
    private double precio;  
  
    public Libro( String unTitulo, String unaEditorial,  
        int unAñoEdicion, String unPrimerAutor,  
        String unISBN, double unPrecio){  
        titulo = unTitulo;  
        editorial = unaEditorial;  
        añoEdicion= unAñoEdicion;  
        primerAutor = unPrimerAutor;  
        ISBN = unISBN;  
        precio = unPrecio;  
    }  
    ...
```

...

```
public double getMontoIva(){  
    return precio*0.21;  
}
```

```
public double getPrecioFinalConIva(){  
    return precio + getMontoIva();  
}
```

}

Obviando la palabra **this** obtenemos el mismo efecto

# La referencia this. Uso.

- Otro uso: para referirse a las v.i.s del objeto dentro de un método/constructor, que posee parámetros con igual nombre que las v.i.s del objeto.

```
public class Libro {  
    private String titulo;  
    private String primerAutor;  
    private String editorial;  
    private int añoEdicion;  
    private String ISBN;  
    private double precio;  
  
    public Libro( String titulo, String editorial,  
        int añoEdicion, String primerAutor,  
        String ISBN, double precio){  
        this.titulo = titulo;  
        this.editorial = editorial;  
        this.añoEdicion= añoEdicion;  
        this.primerAutor = primerAutor;  
        this.ISBN = ISBN;  
        this.precio = precio;  
    }  
    ...  
}
```

```
public void setTitulo(String titulo){  
    this.titulo = titulo;  
}
```

Para referirse a la variable de instancia del objeto usar  
***this.nombreVariableInstancia***

Más información sobre this en:

<https://docs.oracle.com/javase/tutorial/java/javaOO/thiskey.html>